REALTIME
FRAMEWORK

RESTful API

Author	REALTIME.CO – Realtime Framework Development Team
Project	Realtime Cloud Messaging

Review history:

Version	Status	Reviewer	Date
1	Draft	Development Team	07-01-2013
2	Final	Development Team	27-03-2013
3	Update	Development Team	05-12-2013
4	Update	Development Team	23-04-2014
5	Update	Development Team	10-11-2016
6	Update	Development Team	22-03-2017
7	Update	Development Team	31-03-2017

1. Overview

In some languages it is not possible to establish a persistent connection to a Realtime.co server, so in order to support those languages the Realtime Framework team developed a set of RESTful services. All web-services are available in https and http.

2. Messaging API

The web-services parameters are case sensitive. The content type header is **application/x-www-form-urlencoded**.

Operation	Method	Endpoint	Description
server/version	GET	https://ortc-developers.realtime.co/server/2.1?appkey=abcde1 Query string parameters: { appkey : String } Response: var SOCKET_SERVER = "https://ortc-server.realtime.co ";	Returns the server that will provide the best service in that moment. **NOTE:** Before any operation you should always get a server from balancer in order to have the best service possible.
authenticate	POST	http://ortc-developers2-euwest1-S0001.realtime.co/authenticate Parameters: **AT**=String&**PVT**=0 or 1&**AK**=String&**TTL**=Integer > 61&**PK**=String&**TP**=Integer&String=w or r Parameters Description: AT (Authentication Token) = your authentication token PVT (Is Private) = 0 AK (Application Key) = your application key TTL (Time to Live) = 1800 PK (Private Key) = your private key TP (Total Permissions) = 2 Channel=r/w = blue=w&yellow=r Response: - **201**: The authentication token permissions were successfully created - **401**: Not authorized to create permissions, usually this happens when you have an invalid application key	In order to use the ORTC API the user must be authenticated. The way to do this is by making an HTTP POST with some parameters to an ORTC server.

send	POST	http://ortc-developers2-euwest1-S0001.realtime.co/send Parameters: **AT**=String&**AK**=String&**PK**=String&**C**=String&**M**=String Parameters Description: AT (Authentication Token) = your authentication token (if not using PK) AK (Application Key) = your application key PK (Private Key) = your private key C (Channel) = yellow M (Message) = The message to send Response: - **201**: Message successfully sent to the channel - **401**: Invalid application key or the authentication token does not have permission to send messages to the channel *Note: Using the private key parameter (PK) will allow sending messages to any channel. For security reasons this should only be used in server-side use cases.* **Message framing protocol:** You should pre concatenate a string to every message in order for the message to be correctly processed in other API's. That string should contain the following format *12345678_1-1_*. The first part is a random string with 8 characters that will identify your message. The second part is the current message part. The third part is the total number of parts the message contains. Each part should have no more than 800 bytes. Here is an example: 81e070f0_1-1_This is the message	It is possible to send messages without the ORTC API, just using a RESTful web service. The way to do this is by making an HTTP POST with some parameters to an ORTC server.
publish	POST	http://ortc-developers2-euwest1-S0001.realtime.co/publish Parameters: **AT**=String&**AK**=String&**PK**=String&**C**=String&**TTL**=Integer&**M**=String Parameters Description: AT (Authentication Token) = your authentication token (if not using PK) AK (Application Key) = your application key PK (Private Key) = your private key C (Channel) = yellow TTL (Time to Live in secs) = 60 M (Message) = The message to send Response: - **201**: Message successfully sent to the channel. The unique message sequence id is returned in the response body - **401**: Invalid application key or the authentication token does not have permission to send messages to the channel *Note: Using the private key parameter (PK) will allow sending messages to any channel. For security reasons this should only be used in server-side use cases.* See the send method for the message framing protocol.	Publishes a message with guaranteed delivery mode

sendbatch	POST	http://ortc-developers2-euwest1-S0001.realtime.co/sendbatch Parameters: **AT**=String&**AK**=String&**PK**=String&**C**=String&**M**=String Parameters Description: AT (Authentication Token) = your authentication token (if not using PK) AK (Application Key) = your application key PK (Private Key) = your private key C (Channels, comma delimited) = yellow, red, blue M (Message) = The message to send Response: - **201**: Message successfully sent to all channels passed - **401**: Invalid application key or the authentication token does not have permission to send messages to one of the channel *Note: Using the private key parameter (PK) will allow sending messages to any channel. For security reasons this should only be used in server-side use cases.* See the send method for the message framing protocol.	Sends a message to several channels

3. Presence API

Operation	Method	Endpoint	Description
Enable	POST	**http://ortc-developers2-euwest1-S0001.realtime.co /presence/enable/<appkey>/<channel>** Parameters: { privatekey : String, metadata : boolean } Response : String	Enables the presence feature for a specific channel of an ORTC application (appkey). If metadata is true then all subsequent Presence calls will return the connection's metadata for each channel subscriber. If a wildcard sub-channel is used (e.g. channel:*) every sub-channel belonging to channel will have the presence feature enabled automatically

Presence	GET	**http://ortc-developers2-euwest1-S0001.realtime.co /presence/<appkey>/<token>/<channel>**
		
		Response: {
		subscriptions : Number,
		
		metadata : {}
		}
			Gets the current presence data for a specific channel.
			
			The <token> used must have the P (presence) permission set for the requested channel.

Disable	POST	**http://ortc-developers2-euwest1-S0001.realtime.co /presence/disable/<appkey>/<channel>** Parameters: { privatekey : String } Response: String	Disables the presence feature for a specific channel.

4. Mobile Push Notifications API

Note: in order to use the mobile push services you should always use the following endpoint, https://ortc-mobilepush.realtime.co/mp. The content type header is **application/json**.

Operation	Method	Endpoint	Description

List topics	POST	**https://ortc-mobilepush.realtime.co/mp/listtopics**
		
		Body: {
		applicationKey : String,
		privateKey : String,
		nextToken : String [optional]
		}
		
		Response : {
		topics : [String],
		nextToken : String
		}
			Retrieves the subscribed topics (aka channels)
			
			When there are no more topics to be retrieved nextToken will be null. Otherwise a new request should be performed using the returned nextToken parameter to get the next set of topics.

Get Topic Devices	POST	**https://ortc-mobilepush.realtime.co/mp/topicdevices/{topic}**
		
		Body: {
		applicationKey : String,
		privateKey : String,
		nextToken : String [optional]
		}
		
		Response : {
		tokens : [String],
		nextToken : String
		}
			Retrieves the topic (aka channel) registered device tokens.
			
			When there are no more device tokens to be retrieved nextToken will be null. Otherwise a new request should be performed using the returned nextToken parameter to get the next set of device tokens.

Check if topic is subscribed	POST	**https://ortc-mobilepush.realtime.co/mp/isTopicSubscribed/{topic}**
		
		Parameters: {
		applicationKey : String,
		privateKey : String
		}
		
		Response : boolean
			Checks if a topic (aka channel) been subscribed using push notifications.

Publish Custom Notification on a single channel	POST	**https://ortc-mobilepush.realtime.co/mp/publish**
		
		Parameters: {
		applicationKey : String,
		privateKey : String,
		channel : String,
		message : String,
		payload : Object [optional],
		apns: Object [optional],
		gcm: Object [optional]
		}
		
		Payload example:
		
		"payload" : {
		"sound" : "default",
		"badge" : "1",
		"extra" : "something"
		}
		
		Response : null
		
		Invalid Request : { exception : String, message : String }
		
		Note: Sending to a non-existent channel will generate an invalid request
			Sends a custom notification to the specified channel. You can specify a custom payload and alert message.
			
			The apns and gcm body parameters can be used to send raw notification payloads directly to the underlying APNS and GCM/FCM mobile platforms (using the those platforms specific formats).
			
			Learn more about the plattform-specific raw payload feature at https://framework.realtime.co/blog/ios10-push-notifications-support.html

Publish Custom Notification on several channels	POST	**https://ortc-mobilepush.realtime.co/mp/publishbatch**
		
		Parameters: {
		applicationKey : String,
		privateKey : String,
		channels : Array,
		message : String,
		payload : Object [optional],
		apns: Object [optional],
		gcm: Object [optional]
		}
		
		Payload example:
		
		"payload" : {
		"sound" : "default",
		"badge" : "1",
		"extra" : "something"
		}
		
		Response : null
		
		Invalid Request : { exception : String, message : String }
			Sends a custom notification to the specified channels (defined as an array of string), with a maximum of 100 channels.
			
			For more details about the apns and gcm parameters see the publish endpoint description.